Activation Functions in Neural Networks (Reformatted Table)

Activation	Formula / Concept	Motivation (Conceptual)	Advantages	Disadvantages	Computational Aspects	Typical Use Cases
Sigmoid	$f(x)=\\frac{1}{1+e^{-x}}$	Models smooth probabilistic firing behavior of neurons.	Bounded output (0,1); interpretable as probability.	Saturates for large $\lvert x\rvert$ → vanishing gradients; not zero-centered.	Involves exponential computation (moderate cost).	Binary classification output layers, logistic regression.
Tanh	$f(x)=\\tanh(x)$	Center output around zero to accelerate convergence.	Zero mean; smoother gradient than sigmoid.	Still saturates at large $\lvert x\rvert$; vanishing gradients persist.	Uses exponential functions; moderate cost.	RNN hidden layers, older feedforward networks.
ReLU	$f(x)=\\max(0,x)$	Mimics neuron firing when signal is positive; induces sparsity.	Fast and efficient; avoids vanishing gradient for x>0.	Dead neurons for negative inputs; unbounded output.	Just a comparison; extremely cheap computationally.	Default for CNN/MLP hidden layers.
Leaky ReLU	$f(x)=\\max(\\alpha x, x)$	Preserve gradient flow for negative region.	Prevents dead neurons; simple extension of ReLU.	Adds small bias for negative activations.	One multiplication extra compared to ReLU.	Deep CNNs, GANs, general-purpose hidden layers.
PReLU	Learned slope α	Learn nonlinearity from data dynamically.	Adaptable flexibility improves representation power.	Adds learnable parameters; potential overfitting.	Slightly higher compute and memory than ReLU.	ResNet and CNN variants.
ELU	x if $x>0$; $\\alpha(e^x-1)$ if $x\\le0$	Push activations' mean toward zero; smoother than ReLU.	Faster convergence; avoids dead neurons.	Requires exponential computation; sensitive to α.	Higher compute cost than ReLU.	Deep CNNs, MLPs needing smooth activation.
SELU	Scaled ELU (λ, α)	Enforce self-normalization property within layers.	Stable activations without BatchNorm.	Requires specific initialization (LeCun Normal).	Slightly higher compute cost.	Self-Normalizing Networks.
Swish	$x\\sigma(x)$	Smoothly gate inputs; continuous alternative to ReLU.	Smooth gradient; good empirical performance.	Slightly slower; requires sigmoid computation.	Uses both multiplication and exp for sigmoid.	EfficientNet, Transformers.
Mish	$x\\tanh(\\ln(1+e^x))$	Smooth, robust ReLU-like activation with better gradient flow.	Improved stability and gradient propagation.	Higher cost; newer, less standardized.	Uses tanh and ln(1+exp()).	CNNs, modern research architectures.
GELU	$x\\Phi(x)$	Probabilistic, smooth gating of activations.	Smooth; combines ReLU-like sparsity with stochasticity.	Slightly slower; complex derivative function.	Requires erf/CDF computation.	Transformers (BERT, GPT, ViT).
Softmax	$f_i(x)=\\frac{e^{x_i}}{\\sum_j e^{x_j}}$	Normalize logits into probability distribution.	Interpretable; suitable for multi-class outputs.	Sensitive to large logits; potential overflow.	Vector exponential and normalization required.	Output layer for multi-class classification.
Linear	$f(x)=x$	Preserve numeric information; no distortion.	Simple and stable; used in regression.	No nonlinearity, limited expressivity.	Minimal cost, identity mapping.	Regression outputs, final layers.

Pedagogical Notes:

- Nonlinearity is essential for hierarchical feature learning.
- ReLU-family dominates modern deep nets due to simplicity and efficiency.
- GELU and Swish are now standard in Transformers and high-performance models.
- Tanh/Sigmoid still appear in RNN gating units.
- SELU requires careful initialization (LeCun Normal).
- Activation choice interacts with BatchNorm and learning dynamics.
- Output activations depend on task: Linear (regression), Sigmoid (binary), Softmax (multi-class).